



UNITED STATES AIR FORCE RESEARCH LABORATORY

Technology for Maintenance Procedure Validation

**Norman Badler
Charles Erignac
Rama Bindiganavale**

**University of Pennsylvania
Center for Human M&S
200 South 33rd Street
Philadelphia, PA 19104**

Patrick J. Vincent

**TASC, Incorporated
2555 University Blvd.
Fairborn, OH 45324**

**Edgar Sanchez
The Boeing Company**

**Kevin Abshire
Lockheed Martin Corporation**

**Jeffrey L. Wampler
Edward S. Boyle
John D. Ianni**

Air Force Research Laboratory

May 2001

20011102 067

Interim Report for the Period January 2000 to January 2001

Approved for public release; distribution is unlimited.

**Human Effectiveness Directorate
Deployment and Sustainment Division
Sustainment Logistics Branch
2698 G Street
Wright-Patterson AFB OH 45433-7604**

NOTICES

When US Government drawings, specifications or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications or other data, is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Please do not request copies of this report from the Air Force Research Laboratory. Additional copies may be purchased from:

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161

Federal Government agencies registered with the Defense Technical Information Center should direct requests for copies of this report to:

Defense Technical Information Center
8725 John J. Kingman Rd., Ste 0944
Ft. Belvoir, VA 22060-6218

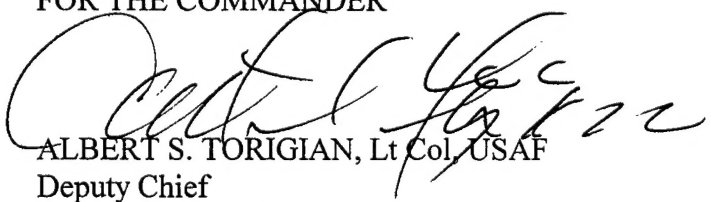
TECHNICAL REVIEW AND APPROVAL

AFRL-HE-WP-TR-2001-0117

This report has been reviewed by the Office of Public Affairs (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER



ALBERT S. TORIGIAN, Lt Col, USAF
Deputy Chief
Deployment and Sustainment Division
Air Force Research Laboratory

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
<small>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.</small>				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE May 2001		3. REPORT TYPE AND DATES COVERED Interim - January 2000 - January 2001
4. TITLE AND SUBTITLE Technology for Maintenance Procedure Validation			5. FUNDING NUMBERS C: F33615-99-D-6001 DO: 08 PE: 62202F PR: 1710 TA: D0 WU: 09	
6. AUTHOR(S) Norman Badler, Charles Erignac, Rama Bindiganavale, Patrick J. Vincent, Edgar Sanchez, Kevin Abshire, Jeffrey L. Wampler, Edward S. Boyle, John D. Ianni				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) TASC, Incorporated 2555 University Blvd. Fairborn, OH 45324 University of Pennsylvania Center for Human M&S 200 South 33rd Street Philadelphia, PA 19104			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory, Human Effectiveness Directorate Deployment and Sustainment Division Air Force Materiel Command Sustainment Logistics Branch Wright-Patterson AFB OH 45433-7604			10. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-HE-WP-TR-2001-0117	
11. SUPPLEMENTARY NOTES AFRL Monitor: Jeffrey L. Wampler, AFRL/HESS, (937)255-7773				
12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for public release: distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This task was performed under delivery order #8 of the Technology for Readiness and Sustainment (TRS) contract (F33615-99-D-6001). The period of research covered activities from January 2000 through January 2001. The primary objective of this research was to explore the feasibility of validating aircraft maintenance procedures using human models. The candidate aircraft maintenance procedures are decomposed from their written form into implied actions to obtain accurate simulation of the procedures. By comparing the procedure language with its corresponding actions, we can easily judge whether procedures as written are safe, logically ordered, and complete. In addition, this task also produced a core set of critical research and development tasks considered essential to overcoming important barriers to the development of an automated Technical Order (TO) validation and verification system or application.				
14. SUBJECT TERMS Technical Orders Human Modeling Human Simulation Computational Linguistics Parameterized Action Representation (PAR) Natural Language Generation (NLG) Technical Order Validation Technical Order Verification			15. NUMBER OF PAGES 40	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

THIS PAGE INTENTIONALLY LEFT BLANK

PREFACE

This task was performed under Delivery Order #8 of the Technology for Readiness and Sustainment (TRS) contract (F33615-99-D-6001). The period of research covered activities from January 2000 through January 2001. The primary objective of this research was to explore the feasibility of validating of aircraft maintenance procedures using human models. The candidate aircraft maintenance procedures are decomposed from their written form into implied actions to obtain accurate simulation of the procedures. By comparing the procedure language with its corresponding actions, we can easily judge whether procedures as written are safe, logically ordered, and complete. In addition, this task also produced a core set of critical research and development tasks considered essential to overcoming important barriers to the development of an automated TO validation and verification system or application.

This project investigated several automated Technical Order validation system components. The principal thrust involved extending a Parameterized Action Representation (PAR) simulator to show task status and human performance failures. We examined the software requirements of both interactive and non-interactive task analyses in order to capture possible task failures. We also studied the automated creation of various PAR data fields from motion capture.

The principal conclusion of this study is that automated TO validation still requires fundamental research work. Some of the research needs include:

Translators that can automatically extract the important actions and PAR characteristics from TO text.

Development of general definitions of PARs for maintenance tasks. These PARs must reference assemblies and parts in a fashion consistent with part nomenclature and Product Data Model (PDM) structures.

Execution of motion generators for complex tasks such as confined reach space planning.

Disassembly planners to insure geometric and spatial access can be used to help sequence maintainer actions; but such planners must be extended to use human reach planning and maintenance access solids as well.

Exhaustive analyses rather than just interactive testing to insure that maintenance tasks can be done by a suitable maintainer population with and without special clothing or protective gear.

Failure reports with sufficient detail to allow visualization of problem areas when exhaustive searches fail. Our demonstration shows that task failures can be detected by monitoring PAR execution and can be illustrated in 3D graphics suitable for task exposition or training.

Interactive component connection and function models to properly mitigate hazards and expose cautions arising from incorrect disassembly procedures or non-geometric operational hazards such as heat, pressure, or radiation.

User interfaces that integrate PDM databases and IETM or similar TO authoring systems with PAR simulation and visualizations.

In general, automated task validation is worthwhile and feasible, but requires additional software tools and integration into the existing Technical Order authoring process.

Table of Contents

Preface	iii
Table of Contents.....	iv
List of Figures.....	v
List of Tables.....	v
Section 1. Parameterized Action Representation Development	1
1.1 Introduction	1
1.2 Simulation Report Status	1
1.2.1 Uniform Interaction Interface	1
1.2.2 Generating Status by System Simulation.....	2
1.2.3 Failure Detection and Handling in PAR	3
1.3 Priorities for Task Simulation.....	3
1.4 Use of Task Analysis Components	6
1.4.1 Non-Interactive Task Analysis	6
1.4.2 Interactive Task Analysis.....	9
1.5 Modify System Components	12
Section 2 ATOV Demonstration	14
2.1 Introduction	14
2.2 Scope	14
2.3 Maintenance Task.....	15
2.4 System Architecture.....	17
2.5 Environment Modeling.....	17
2.6 Agents and Physical Objects in PAR.....	18
2.7 Parametric Action Representation	18
2.7.1 Connector Disconnection.....	19
2.7.2 Point-to-Point Reach.....	20
2.7.3 Power Supply Extraction	22
2.7.4 Other Complex Actions	23
2.8 Failures	23
2.9 Multi-Modeling.....	24
2.9.1 Geometry Observers	24
2.9.2 Assembly Graph	25
2.9.3 Environmental Model	25
2.10 Scenarios.....	26
2.11 Conclusions and Recommendations	27

Appendices	29
A1 LSA Task Narrative (Sample)	29
A2 Assembly Graph Models	29
A2.1 General Models.....	30
A2.2 Maintenance Task Specific Models.....	31
A3 Environmental Model	32
A3.1 General Models.....	32
A3.2 Maintenance Task Specific Models.....	32
References	34

List of Figures

Figure 1. General view of the F-22 upper left weapon bay.	15
Figure 2. E.W. power supply with electrical connectors (cables not shown) and extended handle.....	16
Figure 3. Front and back view of the E.W. power supply.	16
Figure 4. Jack reaching for the connectors on top of the power supply.....	17
Figure 5. The missile launcher (in transparent brown) over-restricts the technician's access.....	18
Figure 6. t_1_shape and t_1_grasp hand shapes.....	20
Figure 7. The four steps of a Disconnect action	21
Figure 8. Connector annotated with target sites.....	21
Figure 9. Power supply extraction	22
Figure 10. Out of Reach Connector	23
Figure 11. Staggered Connectors.....	24
Figure 12. Assembly Graph.....	25
Figure 13. Warning spheres indicating contaminated hydraulic disconnects	26

List of Tables

Table 1. Research Directions	5
Table 2. Priorities for Task Simulation.....	6
Table 3. Non-Interactive Task Analysis with Success and Failure Information.....	9
Table 4. Interactive (VR) Task Analysis with Success and Failure Information.....	11
Table 5. Physical objects in PAR and associated status	18
Table 6. Preparatory specifications for the Extract action.	23

THIS PAGE INTENTIONALLY LEFT BLANK

Section 1. Parameterized Action Representation Development

1.1 Introduction

Since maintenance instructions are authored by people for aircraft maintainers, reducing errors and instruction update costs are desirable Air Force goals. The Automated Technical Order Validation (ATOV) project seeks to use human models to validate that instructed tasks *can* be performed under typical maintenance conditions. Several human form modeling and animation systems exist that are typically used interactively to test human factors and visualize maintenance tasks. In a series of projects, we have examined the issues surrounding the direct interpretation of maintenance instructions as animated tasks. A recent Air Force report (Badler, 00) documented many of the issues involved in maintenance task simulation and validation. One outcome of these studies is a *Parameterized Action Representation* (PAR) to support language-based task understanding, analysis, and visual simulation.

A prototype implementation of PAR and its supporting language interface and graphics simulation engine was constructed as part of this research effort. During the software development phase, several important areas were left unexplored. In this report, we cover some of these issues and advise of the benefits and efficiencies in taking them to implementations. In particular, we cover 1) adding reports of simulation status to enhance user understanding of PAR interpretations, 2) priorities for task simulation software development, 3) a comparison of interactive and non-interactive task validations, 4) a discussion of how PAR information fields might be effectively generated, and 5) general conclusions and recommendations for the next steps in ATOV research.

1.2 Simulation Report Status

Simulation status can be presented under different forms to the user. Although most of the simulation is conveyed through the virtual environment's visual display, associated information may be displayed textually or with schematics. This data, which comes from various simulation domains, must be kept coherent and presented accordingly. Each domain is modeled in a specialized simulation engine. These engines run in parallel to deliver a multi-domain simulation to the user.

The two core domains of the ATOV framework are human factors and maintenance task execution. They are respectively modeled with a human centered virtual environment, such as EAI Jack, and the Parameterized Action Representation (Badler,00). Other domains are relevant to many maintenance tasks. For example, an environmental domain is required to model the handling of hazardous materials and their impact on maintenance personnel and their work environment. Also, many tasks require operating on only partially active systems for the purpose of testing them or for other practical reasons. Some sort of general-purpose simulation facility is a logical complement to ATOV's architecture.

In the remainder of this sub-section we will present our approach to a general-purpose system simulator. First, we address the problem of keeping multi-domain simulations synchronized. Then we will discuss simulation status presentation. Finally we discuss adding failure detection and failure handling in PAR.

1.2.1 Uniform Interaction Interface

In the ATOV framework most of the simulation status is conveyed through the geometrical appearance (shape, position, orientation, and color) of objects. These objects can be modeled in other non-geometric domains (PAR, physics-based) in order to generate relevant behaviors such as functional modes, fluid flows, or heat transfer. During the simulation, these models must be kept synchronized through their common or corresponding properties. We will call the set of models representing an object in various domains a multi-model. This multi-model will be said to be *coherent* if its underlying models are synchronized.

In virtual environments as in the real world, most human-device interactions are generally done by directly manipulating physical objects. The effect of these interactions can have associated consequences in non-geometric domains.

For example, moving the geometric model of a plug into the model of a socket should translate into an electric connection between their respective electrical models. Furthermore, if the insertion was performed through a PAR action, the PAR engine must be informed that the insertion is complete in order to terminate the action.

Modeling such interactions across many domains and with various performers (user or autonomous agents) can become intractable. To simplify multi-modeling and guarantee coherent behavior, we advocate creating a unique interaction interface between a physical object and the rest of the environment. This interface should encapsulate the multi-model synchronization mechanism to make it transparent to the user.

In the maintenance domain, the physical behavior of an object depends on its assembly structure. In other words, most object behaviors are ultimately geometrically driven. In our connection example, it is the geometric relationship between the plug and the socket (i.e. contact) that creates the electrical connection.

In some cases, causality is reversed. For example, the position of a gauge is driven by the pressure it measures. Multi-model synchronization is feasible as long as causal dependencies between domains are well defined and if feedback loops are stable.

The uniform interaction interface approach has two benefits. First, the status of an object is always coherent across modeling domains. This includes the initial state. Second, two different means of interaction, such as direct user manipulation and procedural agent manipulation, are guaranteed to have the same effects.

In section 2.9, the uniform interaction approach is implemented in the ATOV demonstration.

1.2.2 Generating Status by System Simulation

General-purpose simulators are routinely used to generate aircraft systems status for various simulation tasks. Aircraft maintenance simulation is one of them. However, unlike other applications, it requires altering simulation models as the virtual technician assembles or disassembles a component from an aircraft system. Few simulators can update their simulation models on the fly. Those simulators that can do not have the adequate modeling capabilities to capture aircraft systems. We developed a simulator as an extension of PAR to meet these requirements.

The PAR simulator (Erignac,01) was designed to simulate *assemblable* systems of moderate complexity. It uses hierarchical finite state machines to enable compact device models. This paradigm, also known as Statecharts (Harel,87), is widely used to model weapon systems. It is particularly adequate to handle the extra complexity of assemblable models. A statechart is a graph where the nodes are states and the edges are transitions. Each node can contain its own graph. This hierarchical structure captures adequately the operating modes and sub-modes of complex devices. In addition, each state contains quantitative constraints to model the dynamic behavior of a device in the corresponding mode.

During the simulation, an expert system uses modeling rules to instantiate and connect model components. It keeps the models up to date to reflect the alterations of the virtual technician. This automated model building technique was developed in the field of qualitative simulation. It allows maintaining a qualitative interpretation of the model that matches the abstraction level of PAR. This simplifies interfacing between action and system models.

The model building rules are declared in the models themselves. They encode model-building expertise in the form *if condition then consequences*. The condition is based on the prior instantiation of certain model instances, such as device components, and on the state of simulation, such as a fluid level or a tank pressure. The consequences is a set of constraints to enforce or models to instantiate when the rule fires. The rule system is *monotonic*. When the firing condition of a rule ceases to hold the rule's consequences are withdrawn from the global simulation model (see sections 2.9.2 and 2.9.3 for examples).

1.2.3 Failure Detection and Handling in PAR

For various reasons, maintainer actions may fail when simulated. The failure may be due to a faulty design or faulty instructions, or may be due to human factors limitations of the technician. In this work we assume that the design is not the cause, although the same simulation framework could be used during design stages to assess task feasibility. Here we will focus on failures caused by instruction orderings, collisions with interfering components, exposure to potentially hazardous substances or pressures, or human reach limitations.

Using PAR, we will generally define failure as the occurrence of an explicitly expected failure (termination) condition or the inability to satisfy the applicability conditions or preparatory specifications of an action. The PAR architecture handles such failures during the execution of an action. During execution, all failures are detected and reported by motion generators to the process manager within the agent process. The failure specified by the motion generator contains the following information:

- Failure code (represented as a string)
- Pointer to the instantiated PAR unit which failed
- Relevant information stored in compact structure to help in failure recovery at a later stage

The process manager does error recovery and handling in two ways:

- The process manager freezes the queue manager such that no other actions are popped from the highest level of the instantiated PAR queue. It also informs the agent process of a failure and passes it the above information. The agent process can now make a decision about the failure recovery process. It could restart the queue as is or by flushing the entire action queue of the agent or by inserting a new high priority action into the queue.
- The process manager also takes care to abort all actions that lie in the path from the failed primitive action up to the highest complex action (which hierarchically invoked the failed primitive action). Thus, for example, if deep within a high level complex action, a primitive action fails, then all actions up to the complex action are automatically aborted. Alternately, different action(s) can be selected anytime instead of aborting all the actions.

This mechanism allows for a very general approach towards failure detection and recovery. Based on the simulation, the agent process could also invoke the planner to generate a new path or set of actions using the information passed from the motion generator.

1.3 Priorities for Task Simulation

Effective automated validation will require computational components to check task features that are typically difficult for the TO author to perform without either trusting other sources (LSAR) or going to the aircraft for a hands-on test. These components include:

- confined space reach planning
- disassembly planning
- reasoning about the behaviors and functions of complex devices
- modeling operational hazards such as component temperatures, pressures, radiation, exposure danger, etc.

We established priorities by analyzing the sorts of tasks that maintainers are being asked to do and by the relative difficulty the TO author has in assessing the critical issues in a particular task. For example, in the uplock hook case we studied in D.O.#1, access is not as much of an issue as the object structure (function) is: if the attachment bolts are removed the device falls apart (an undesirable occurrence). In other situations, access may be more critical; while in others preventing exposure to or release of hazardous substances may be paramount. Methods for computationally modeling these components and making them accessible during and for validation were studied.

The results of our analyses are summarized in Tables 1 and 2 below. Table 1 analyzes the requirements for task validation based on the four features: confined reach planning, disassembly planning, reasoning about the behaviors and functions of complex devices, and modeling operational hazards. For each feature, we first analyzed the difficulty of mentally (that is, at a desk with only visual and textual displays) determining the feasibility of a task that required this type of reasoning. Then, we assessed the difficulty of producing algorithms and software to perform the same sort of analyses automatically and determined if COTS or at least known software solutions might be available. Within the software solution scope, we then tried to isolate the hard problems or issues that could frustrate a near-term software solution or otherwise make a complete software solution expensive or protracted. Table 1 summarizes our findings and helps expose the relative costs and benefits of attacking these validation system components. In Table 2, we extend the analysis of the same four validation features to venture some estimates on software implementation costs. This analysis includes a listing of the implementation issues that must be addressed, assumptions that are made about the scope of an adequate implementation, factors beyond the immediate control of the Air Force or its current Delivery Order partners, and any dependencies between the features themselves. We use this analysis to estimate a relative priority for a software implementation of each feature based on these observations.

In our estimation, the greatest benefits will come from investments in confined reach space planning. Given that maintainers are limited to accessing what components they can reach, and that reach is the hardest task to judge from a visualization, we believe that effective computational tools for evaluating accessibility across a range of typical maintainer body sizes is the highest priority for automated task validation. While existing human form models allow a user interactive access to do reach analyses, they do not effectively represent the confinement constraints: the user is responsible for guiding the human form model into an appropriate configuration. In general this is sufficient for straightforward cases, but not at all efficient if the reach is feasible but requires an unusual posture. Determining that *no* posture will be effective may be an arduous, tedious, and perhaps redundant burden for a TO author, especially if the author trusts that the engineering design team has already validated the reachability aspects of the task. Confined reach analysis is complicated by the variety of strategies maintainers employ to access components. While interactive analyses may serve the validation well by permitting the user to suggest likely postures in unusual cases, these may also serve to guide a more automated search or anthropometric accommodation analysis.

The second highest priority validation feature we note in Table 2 is *disassembly planning*. While various disassembly planners have been prototyped, none incorporate the human maintainer's constraints (in any general fashion) into a proposed assembly step ordering. In the various studies we have done for the Air Force we established that a disassembly order based purely on the geometry and connections of the modeled components will fail to take into account maintainer accessibility (thus the dependence on confined reach planning as noted in Table 2), device contents (fluids), and potentially hazardous situations (electrical current, heat, radiation). Existing software tools for disassembly planning may be extended to include or at least consider these factors as ordering constraints. We do not know of any such comprehensive planning system at this time. Developing one based on extensions of a purely geometric planner would serve as a useful automated validation aid by incorporating exactly those parts of task analysis that the instruction author is obligated to consider.

To include functional considerations and hazard conditions, automated validation tools will need access to comprehensive models of system structure and behavior, as noted above. The TO author's training includes knowledge of system functions, operation, and hazard conditions, so this area is less critical to automate for ATOV. If the complexity of such knowledge may exceed the training or background of a typical TO author, then investment in automated tools that utilize this knowledge during ATOV could be justified. At the present time, however, we feel that TO authors (in consultation with other members of the engineering design team) are likely to have sufficient experience and resources to take functional, operational, and safety hazard conditions into account manually during task validation.

We will investigate this issue of interactive versus automatic (i.e., purely computational) approaches to validating typical tasks next in Section 1.4

	Difficulty of (at desk) decision for TO author	Difficulty of software implementation	Availability of software solutions	Hard problems or issues
Confined Space, Reach Planning	Difficult (hard to visualize maintainer posture and arm trajectories for access and extraction)	Difficult; requires finding feasible reach path and postures of high degree of freedom articulated volumes (human body) and CAD solids (tools and removed components)	Human figure or robot simulations; may be limited to low degrees of freedom reach and manipulation because of computational complexity (exponential growth in cost with degrees of freedom)	Managing anthropometric variability; Modeling hand shapes; Modeling clothing or other gear constraints
Disassembly Planning	Difficult (hard to visualize constraints; need maintenance access solids; guess at tool access)	Difficult; requires detailed CAD model and features	Maintenance access solids in proprietary software systems	Difficult to include human maintainer; May handle only rigid geometries; No understanding of non-geometric system components
Reasoning about the Behaviors and Functions of Complex Devices	May require perusal of schematics; May require functional understanding of complex electronic, fluidic, propulsion, and mechanical systems	Difficult; requires quantitative and qualitative models of system behavior	Engineering simulations pervasive in engineering design, but may not be suitable for interactive semi-qualitative reasoning	Creation of suitable semi-qualitative models from engineering schematics is an unsolved problem: original schematics may be too costly to digitize or use as computational simulations
Modeling Operational Hazards	May require perusal of schematics; can use standard cautions and warnings	Easy if CAD models are tagged with attributes describing contents or operational parameters (temperature, pressure, radiation)	Depends on whether a Product Data Model (PDM) stores appropriate information	If not in PDM it must be inferred from schematics and operating specifications; might be easily inferred from TOs for similar systems; ought to be part of TO author basic knowledge

Table 1. Research Directions

	Time required for prototype and/or useful implementation	Issues Addressed	Assumptions	Uncontrollable factors	Dependencies	Relative priority
Confined Space, Reach Planning	2 person years	Complete human figure degrees of freedom; include automatic posture selection and reach path feasibility	Detailed finger manipulation model unnecessary; use a library of stored handshapes and motion volumes appropriate to task	Requirements for maintainer anthropometric variability accommodation would increase cost by about 50%	Good human figure model with shape and joint range of motion and preferably a good strength model for arbitrary postures	90%
Disassembly Planning	1 person year	Start with CAD component models, produce disassembly sequence respecting maintainer access	Start with existing disassembly planner and extend to include human maintainer and maintenance access solids	Availability of PDM for CAD objects in assembly; otherwise must manually annotate parts	Requires effective implementation of confined reach space planning	80%
Reasoning about the Behaviors and Functions of Complex Devices	2 person years	Build and maintain semi-qualitative models of selected aircraft subsystems	Manually construct semi-qualitative models	Availability of suitable and current digital schematics	Digital schematics could be processed to automate generate of semi-qualitative models	50%
Modeling Operational Hazards	1 person year	Automatically annotate PDM, CAD, or digital schematics with operational hazards	PDM or digital schematics exist	No true understanding of underlying physics; must rely on engineering models as provided	Need PDM for CAD models plus digital schematics or else manual annotation	20%

Table 2. Priorities for Task Simulation

1.4 Use of Task Analysis Components

Task analyses form an essential part of human factors evaluations. For ATOV, we can leverage some human form model software features to help determine task viability. We can consider using these features two ways: non-interactively and interactively. In a non-interactive approach, the software system essentially launches an analysis module and awaits a report on success or failure. In an interactive approach, the burden is placed on the user to iterate over the human model parameters and actions to attempt to satisfy task validation requirements. For purposes of ATOV, *the task must be assumed infeasible until it can be demonstrated that it is performable*. In other words, given the state of the aircraft and the range of capabilities of typical maintainers, the person can move and reach (perhaps with a tool) some identifiable component with sufficient strength to release and extract the part safely. If only some of these conditions are met, the task may be infeasible and the instructions may need to be modified. Task validation fails if any of these component conditions fail.

In the following sections, we will enumerate some of the task analyses that may be needed for ATOV. Then we will look at non-interactive (algorithmic) software requirements to provide useful task failure information. Finally, we will address the efficacy of and problems with interactive approaches (such as Virtual Reality) for similar purposes.

1.4.1 Non-Interactive Task Analysis

When a PAR is executed on a human form model, it expects information back from the sub-action PARs or motion generators if, for example, a reach fails or the agent cannot fit into a space. Ideally, the information

should be returned to the PAR simulator as outlined in Section 1.2.3 (rather than just output as a message on the computer screen) so that the simulator can try alternatives or signal a true validation failure to the TO author. Various task analyses may be needed for TO validation:

- Is the component reachable?
- Is the component graspable?
- Is the component visible?
- Is the component haptically identifiable? (Haptic refers to the felt shape of an object.)
- Is a tool needed to reach, release, or extract the component?
- Does the maintainer have sufficient torque/strength to perform the part release and extraction?
- Does the presence of Arctic or MOP-gear invalidate otherwise performable actions?
- What is the accommodation range of human models that are capable of performing the task?

Different human modeling systems may provide varying levels of response to these queries. Clearly the analysis features needed for human factors during the design phase are nearly identical to the needs for TO validation, the main exception being the source of the task. During the design and evaluation phase a human factors engineer will guide a human model interactively for task feasibility. These analyses may be based on incomplete CAD models as other portions of the design may not be finalized yet. In the TO case the actual instructions are the source of the task and the CAD model should be complete. It may be possible to integrate these two activities by saving the human factors analyses and human form motions in PARs and replaying them during TO validation. These PARs may form an approximate motion path that could dramatically improve the search requirements for constrained space reach planning.

Looking at task analyses from the TO validation perspective provides the software requirements listed in Table 3. For each analysis requirement we outline the algorithmic techniques that could be used for performing that analysis, and comment on any particular features of or constraints on the algorithms. Were software written to perform such an analysis, we indicate what sorts of information must be returned to the user to indicate success or failure of the analysis. A successful result implies validation of the task. A failure can indicate the cause and can also provide other useful information (such as closest reach distance) that can be used to assess tool or assistive device needs. Some analysis failures (such as visibility) may not invalidate the task, but may indicate where a 2D graphic illustration may be appropriately inserted into the instructions.

Software Feature Analyzed	Techniques and Comments	Software Return Information Requirements
Is the component reachable?	Reach analysis must include the whole body shape, postural stability requirements, as well as arm and hand motion. Constrained reach space planning is needed, possibly with a disassembly analysis to order part extractions according to geometric feasibility. As discussed above, these two software components are the highest priority items needed to realize significant benefits with automated TO validation.	SUCCESS: return paths and body configuration FAILURE: return indicators of closest approach point and CAD object set involved in collisions
Is the component graspable?	A library of handshapes for expected grasps augmented by required movement ranges could be geometrically and statically checked for clearance. These libraries could include gloved hands and representative anthropometric sizes and typical movement ranges. In practice, an articulated hand would be used during constrained reach path planning and then the static geometric volume would be used for a grasp and manual manipulation collision test for access and task validation.	SUCCESS: return success, possibly with anthropometric accommodation range FAILURE: return maximum CAD object-to-handshape penetration distance and illustrate with objects and handshape geometry
Is the component visible?	The human form model should provide a visibility assessment. An obscured or partly obscured component may alert the TO author to the need for a graphic with a cut-away view. The visibility determination must account for a reasonable range of accessible viewpoints for the maintainer's eyes.	SUCCESS: component is visible in model's view cone space, and possibly as partly obscured; least obscuring feasible viewpoint should be returned FAILURE: component is tagged as not visible from any feasible view
Is the component haptically identifiable?	Haptic identifiability is not likely to be part of any foreseeable human modeling capability, though human maintainers do rely on feel for obscured task targets.	SUCCESS: component can be reached FAILURE: component can be neither seen nor reached
Is a tool needed to reach, release, or extract the component?	The original human factors analysis may provide this information. If reach or access fails during TO validation, however, the TO author must consider the possibility of using an appropriate tool. While not impossible, it does not seem feasible at present to expect the ATOV software to suggest or invent a suitable tool.	SUCCESS: task is successful with tool in hand; same result as with reach FAILURE: return indicators of closest approach point and CAD object set involved in collisions with both tool and body

Software Feature Analyzed	Techniques and Comments	Software Return Information Requirements
Does the maintainer have sufficient torque/strength to perform the part release and extraction?	This should also be determined during design stage human factors analyses. Present human form modeling systems are probably not well enough developed to do more than roughly estimate strength requirements for complex reach or extraction tasks.	SUCCESS: task is successful; return time line of required joint torques or other strength indicators FAILURE: return body joints overstressed by what amounts and at what times
Does the presence of Arctic or MOP gear invalidate otherwise performable actions?	The simulation may be re-run with suitable human model shape and mobility enhancements. These should also be run during human factors evaluations. Existing human form modeling systems should provide this functionality.	SUCCESS: return paths and body configuration FAILURE: return indicators of closest approach point and CAD object set involved in collisions
What is the accommodation range of human models that are capable of performing the task?	The automated TO validation may need to be iterated over a selected population of specific individuals represented as 3D human form models. While this should also be expected from a human form modeling system, the dependency on accurate and efficient constrained reach planning means that it is unlikely to be provided unless specifically required or developed independently.	SUCCESS/FAILURE: return anthropometric ranges and/or set of individuals tagged as accommodated or not

Table 3. Non-Interactive Task Analysis with Success and Failure Information

1.4.2 Interactive Task Analysis

Most current human form modeling systems have provided the user with an interactive interface for positioning and moving the body. With such a system, certain task analyses such as reach, view, or strength can be performed interactively to establish feasibility. It is relatively easy to establish part visibility. Reachability in confined spaces may be more difficult since the burden of posture determination and collision avoidance is foisted onto the user. By pushing this obligation onto the user, however, certain anomalous or failure conditions may be overlooked. The software may not have the capability of dragging the end effector into a constrained space, so complex reaches may not be properly evaluated. A part that was successfully installed at manufacture time, and whose maintenance access solid was assiduously respected during the design phase, may nonetheless be unextractable due to constraints on the maintainer's posture from other components. In general, human form software systems treat task failures as a user problem: the situation must be remedied via design iteration.

Since these difficulties with existing interactive systems exist, researchers have been tempted to explore interactive manipulation solutions where the user's body is explicitly represented in the 3D world. By using virtual reality systems, the user is expected to see, manipulate, and sometimes even haptically experience a complex 3D environment. Accordingly, we can examine the task analyses requirements in the context of a participant experience in direct 3D observation and manipulation of the aircraft system environment. Table 4 gives a summary of our observations on the effectiveness of virtual reality techniques for ATOV task analyses. It is structured analogously to Table 3 to facilitate comparisons.

VR Software features	Virtual Reality Techniques and Comments	Virtual Reality Software Evaluation
Analysis		
Is the component reachable?	<p>Reach analysis can be based on human postures, but direct feedback from the CAD environment is indirect, e.g. body collisions or penetrations as color changes or sounds. The burden of finding a feasible reach is on the user.</p> <p>The user cannot readily brace against physical supports and this may lead to inaccurate assessments of reachability. <i>(See the discussion in the Conclusion on haptic sensing of body penetrations.)</i> Disassembly analysis may still be needed to order part extractions according to geometric feasibility.</p>	<p>SUCCESS: Object is reached and body configuration is noted.</p> <p>FAILURE: empirical testing fails to achieve goal.</p>
Is the component graspable?	<p>A handpose sensing glove may be worn to access handshape for grasping, but there is no guarantee that the pose will create the necessary force or torque to perform the action.</p>	<p>SUCCESS: the hand can be positioned on the object in a desired handpose.</p> <p>FAILURE: by empirical experiment, no handpose offers a collision-free volume.</p>
Is the component visible?	<p>The user changes posture until the desired component is viewable. The posture throughout the motion must be feasible (penetration-free), and this is difficult to guarantee. <i>(See reachability discussion above.)</i></p>	<p>SUCCESS: component is visible in user's view space.</p> <p>FAILURE: by empirical experiment component cannot be seen from any achievable posture.</p>
Is the component haptically identifiable?	<p>Haptic feedback may be used to provide reaction forces but present feasible technologies do not permit enough freedom in the location of the end effector in space.</p>	<p>SUCCESS: component can be felt with haptic feedback glove.</p> <p>FAILURE: component cannot be portrayed haptically (e.g. a liquid)</p>

VR Software features Analysis	Virtual Reality Techniques and Comments	Virtual Reality Software Evaluation
Is a tool needed to reach, release, or extract the component?	Either the user must physically hold the tool and the system must know that, track it appropriately, and compute collisions with the CAD environment; or the user must grasp a virtual tool. In the former case the actual tools are needed and these may confound real-time motion capture of the subject (due to metal interference with the electromagnetic fields used for pose sensing). In the latter case the feeling of weight and inertia is missing and the motion of the tool must be inferred from the user's movements: this often yields a very jerky, irregular motion that is difficult and tiresome to control.	SUCCESS: task is successful with tool in hand; same result as with reach. FAILURE: empirical testing fails to achieve goal.
Does the maintainer have sufficient torque/strength to perform the part release and extraction?	Force feedback is presently only realistic in highly restricted areas of space due to portability and weight limitations of haptic feedback hardware. Full body force reflective robotic devices are not realistic due to expense, danger, and availability.	SUCCESS: not likely to be reliably ascertained. FAILURE: not likely to be reliably ascertained.
Does the presence of Arctic or MOP gear invalidate otherwise performable actions?	The user may wear the actual gear and perform the required tasks. The gear must be available, must be donned, and if metal, may interfere with electromagnetic pose sensing hardware.	SUCCESS: Object is reached and body configuration is noted. FAILURE: empirical testing fails to achieve goal.
What is the accommodation range of human models that are capable of performing the task?	Multiple experimental sessions must be performed with representative individuals attempting all the required tasks.	SUCCESS/FAILURE: tag individuals as accommodated or not

Table 4. Interactive (VR) Task Analysis with Success and Failure Information

Based on this analysis, we would not recommend that Virtual Reality systems be relied upon for ATOV. VR systems can play an important role in training but they appear awkward for assessing task feasibility or determining the range of accommodated maintainer populations. We believe that robust algorithms for confined reach space planning and iteration of this algorithm over candidate maintainer anthropometric samples is a more cost effective method of achieving ATOV.

1.5 Modify System Components

Although the preceding section shows that non-interactive analyses are in general more effective in establishing ATOV, there is a role that VR systems can play. The PAR provides a computational analog of the TO that drives a human figure model. Where does this PAR come from? Since ATOV requires executing maintainer actions, we must justify that the information fields in a PAR may be established through reasonable and efficient means. The purpose of this part of the ATOV study was to determine how the PAR data fields could be set or modified to create a maintainer action database.

A major motivation for our work was the belief that VR techniques might be used to establish the *motion* component of a PAR. The PAR action could therefore be *performed* rather than tediously designed or programmed. *Notice that this is not the same as validating a particular task action's effectiveness as analyzed in Section 1.4; rather, we are creating the definitions of the actions themselves.* A key idea is that the VR approach must be augmented by algorithms that determine what features of a particular action are salient and thus generalizable to other maintainers and other environmental contexts. In short, a performance is used to *teach* the system what a particular PAR does. We describe our approach in this Section.

We have built a prototype PAR creation system that allows needed data fields to be supplied through a variety of user interfaces. We use graphical interfaces, natural language, and motion capture from a live performance. Given each field of the PAR, we experimented with what sorts of inputs (GUI, language, motion capture) are best for establishing needed information. In general, attributes that do not refer to continuous aspects of a PAR are easily set via a GUI or language inputs. The movement itself must be created through a human modeling and animation system or else obtained from motion captured from a live subject. The case of scripting motions directly is well established in the animation software community and in common human form software systems. An additional feature of the PAR creation system is the means to search for existing PARs for re-use. This feature reduces the input required while producing more efficient code.

We explored the possibility of using real subjects performing a task and then creating the appropriate PAR as automatically. A system was created for this purpose called Captured Parametric Action Representation (CaPAR) (Bindi.,00). CaPAR analyzes the 3D motion inputs from a subject wearing electromagnetic sensors. It computes significant changes ("zero crossings" or sign changes) in acceleration of end effectors. When these events occur in proximity to objects in the environment, CaPAR considers them as significant aspects of a movement. Such events become constraints on the agent's motion that can be stored in a PAR and later interpreted as reaches or alignments when executed on another simulated agent, possibly of a different body size. In this fashion the PAR motion information can be collected in an agent-neutral fashion and reused in other contexts with other agents.

Along with CaPAR, methods for obtaining data for the important fields of a PAR are described below.

- **Agent:** The agent executes the action. An agent, considered as a special type of object, has a number of properties and is stored as part of the hierarchical object database.
- **Objects:** The object type is defined explicitly for a complete representation of a physical object and is stored hierarchically in a database. Each object in the environment is an instance of this type and is associated with a graphical model in a scene graph. The state field of an object describes a set of constraints on the object that leave it in a default state. The object continues in this state until a new set of constraints is imposed on the object by an action that causes a change in state. The other important fields are the reference coordinate frame, a list of grasp sites and their purpose, and intrinsic directions (top, front, etc.) defined with respect to the object. In our example, the walking action has an implicit floor as an object, while the turn action refers to the handle. These object attributes are created through menus or a GUI.
- **Applicability Conditions:** The applicability conditions of a PAR specify what needs to be true in the world in order to carry out an action. These can refer to agent capabilities, object configurations, and other unchangeable or uncontrollable aspects of the environment. The conditions in this Boolean expression must be true to perform the action. For "walk," one of the applicability conditions may be "Can the agent walk?" If conditions are not satisfied, the action cannot be executed. The applicability

conditions of an action have to be entered explicitly by the user through language or programming interfaces and cannot be recognized or generated automatically by the CaPAR system.

- **Preparatory Specifications:** This is a list of <CONDITION, action> statements. The conditions are evaluated first and have to be satisfied before the current action can proceed. If the conditions are not satisfied, then the corresponding action is executed; it may be a single action or a very complex combination of actions, but it has the same format as the execution steps. In our example, one of the conditions to be checked could be *posture(agent) = stand* and the corresponding action could be (*stand*, agents: ("Jack")). If the agent is in a sitting posture or prone posture, then the action causes him to change to the standing posture. The preparatory specifications can be generated by the CaPAR system at run-time or can be manually set by the user.
- **Execution Steps:** A PAR can describe either a primitive or a complex action. The execution steps contain the details of executing the action after all the applicability and preparatory conditions have been satisfied. If it is a primitive action, the underlying Pat-Net for the action is directly invoked. A complex action can list a number of sub-actions that may need to be executed in sequence, in parallel, or as a combination of both. A complex action can be considered done if all of its sub-actions are done or if its explicit termination conditions are satisfied. The execution steps for both primitive and complex actions can be generated by the CaPAR system or can be manually specified by the user by providing the names of the appropriate sub-action PARs.
- **Core Semantics:** The core semantics represent a PAR's primary components of meaning and include the following:
 - **Motion:** This specifies the object that is being moved and the type of motion: rotation, translation, or both. It also specifies if this is a causal motion. The type of motion to be executed can be specified through natural language.
 - **Force:** This points to the affected object and indicates the force or torque amount and point of contact.
 - **Path:** This contains information on the location of the object at the beginning or the end of the motion and the directional changes that model the approximate path of the motion. The path information can be generated automatically by the planner component of the PAR system.
 - **Purpose:** This specifies the state or condition that will be achieved as a result of this motion. It also points to the PARs that are either generated or enabled during the course of or at the end of the motion. It is best input as a natural language statement. Sometimes it is implicit in the higher level actions: for example, "Remove the UpLock Hook" implicitly provides the purpose for the sub-actions that achieve this goal.
 - **Termination Conditions:** This is a list of conditions that, when satisfied, complete the action. These can be generated from natural language or from motion capture. From natural language, the termination condition can be determined from the main verb or attached clauses. For example, the "Remove UpLock Hook" action is terminated when the remove action is successful. From motion capture, the CaPAR system generates this automatically at run-time.
 - **Duration:** This specifies the duration of the motion. This can be specified either through the GUI or from natural language.
 - **Manner:** Manner specifications describe the way in which an agent carries out an action. The CaPAR system abstracts and derives an agent's style or manner of doing the action. It may be retained or discarded at the user's discretion. The manner to be used for a specific instantiation can also be specified by natural language.
 - **Post Assertions:** This is a list of statements or assertions that are executed after the termination conditions of the action have been satisfied. These assertions update the database to record the changes in the environment. The changes may be due to direct or side effects of the action. These are provided through the GUI or language interface.

Section 2

ATOV Demonstration

2.1 Introduction

The ATOV demonstration is meant to display the ability to validate maintenance instructions by simulating their execution in a virtual environment. A virtual human is placed in a virtual setting corresponding to the input conditions of the maintenance task. The user translates each step of the instructions into high-level commands that are executed by the virtual human. The simulation system generates the failures, hazardous, and unwanted situations deriving from the virtual human carrying out the commands. A procedure can be deemed valid if no such failures or hazards occur.

The selected scenario is the removal of an electronic warfare power supply from an F-22 jet fighter. It involves reaching in confined spaces, disconnecting electrical and hydraulic lines, releasing fasteners, and extracting the unit from the airframe.

Among all the cognitive and physical simulation capabilities required to support such a validation system, the demonstration focuses on:

- Translating high-level maintenance instructions into the set of low-level actions performable by the virtual human.
- Reporting action failure.
- Generating simulation status to monitor the progress of the maintenance task and detect hazardous situation.

For this demonstration, other required capabilities such as motion planning were emulated.

2.2 Scope

ATOV focuses on the interactive execution of high-level maintenance instructions by a virtual human. This agent is expected to use its “prior knowledge” to generate the sequence of appropriate low-level actions, such as selecting the appropriate tool and sequencing sub-actions according to accessibility conditions, to carry out its instruction. In case the instructions are incorrect or if its knowledge is incomplete appropriate failures should be reported. These failures can be based on the impossibility of performing a task or the occurrence of a hazard. This demonstration will show how the Parameterized Action Representation (PAR) can be used to model the virtual human’s knowledge and how the agent uses it to execute a maintenance task. It will also show PAR’s ability to detect failures and report them.

The demonstration is implemented with the PAR system running within the Engineering Animations Inc. (EAI) Jack-2.2 virtual environment.

In order to take complex decisions, such as path or disassembly planning, the agent must use dedicated “skill” modules for specific reasoning tasks. The PAR system coordinates their use and supervises action execution. At the time of this project, the skill modules have not been developed. Their function will be emulated. This demonstration is a means of identifying and prioritizing the specific skills that must be built into the agent (see section 1.2).

We will emulate motion planning capabilities with sequences of point-to-point arm reaches between predetermined locations in space. These landmarks are attached to objects of interest (connectors, power unit, etc.). The virtual human uses them to approach, grasp, and move about these objects. As a consequence the human’s motion will be robotic in nature and will lack the natural smoothness and directed movement (directed) conferred by a motion planner.

Contact interactions between a human and a solid object, or between two solid objects, are one of the most basic features of the physical realm. Computational cost of current technology does not allow real-time simulation of contact interaction in complex environments such as an F-22 weapon bay. Also, contact reaction depends on the nature of the colliding objects and the intent of the virtual human. A physics-based

collision handling between two rigid objects is straightforward to implement. On the other hand a realistic collision between a deformable object, such as a virtual human, and a rigid object is currently impossible to model in real-time. This would involve compressing the deformable object along the contact area. The F-22 scenario requires reaching in confined spaces where the technician's hands and arms will systematically touch, slide along, or rest on parts of the airframe. We will not model these contacts. As a consequence polygon interpenetration will be observed, but should not be construed as a design error. Other interpenetrations might be observable due to the lack of motion planning.

We will limit the simulation to detecting collisions between the human and certain elements of the scene (i.e. the power supply's connectors). This will give us the opportunity to show the importance of the disassembly planning skill.

2.3 Maintenance Task

The demonstration depicts a virtual human performing a maintenance task on an F-22 aircraft. The task pertains to the removal of an avionic component, an Electronic Warfare (E.W.) power supply, from the upper left weapons bay of the airplane. We used CAD files and the task's Logistic Support Analysis (LSA) record (see 2.11) provided by Lockheed-Martin as our main technical documentation. This scenario was initially used in a design maintainability demonstration (Barron,97). This study showed that the proposed power supply's location and shape raised many accessibility problems.

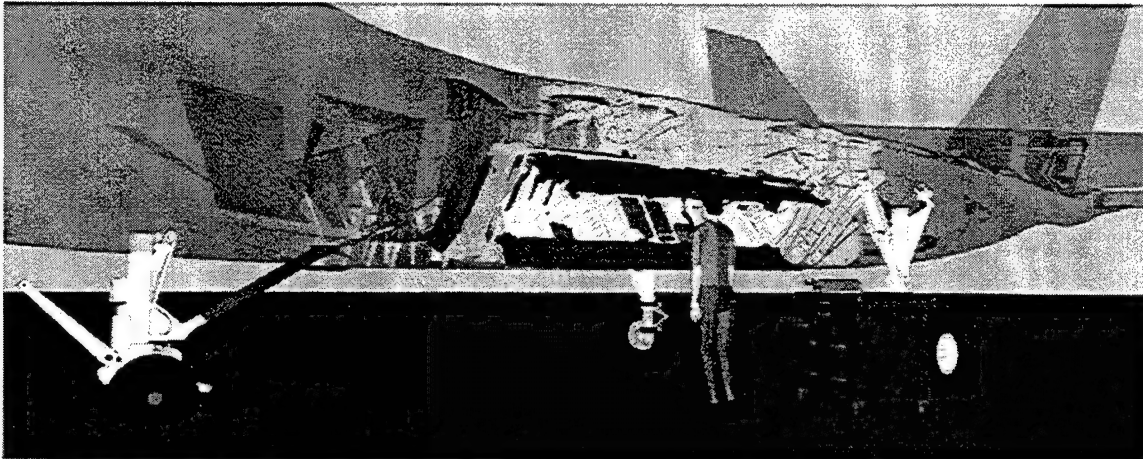


Figure 1. General view of the F-22 upper left weapon bay.

The E.W. power supply rests inclined at 30° from the vertical between two bulkheads near the center of the weapons bay. It is the size of a big shoebox (17.72×7.10×5.12 in. or 45×18×13 cm) and weighs 25 pounds (11.3 kg). A handle is located on the front. It can be locked in two positions: stowed and extended. The handle is used to extract and carry the unit and is kept stowed otherwise.

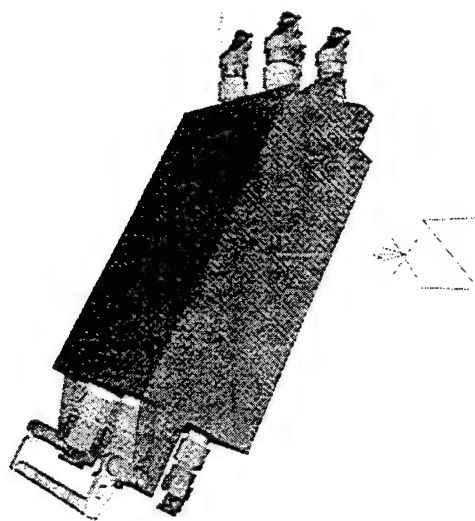


Figure 2. E.W. power supply with electrical connectors (cables not shown) and extended handle.

The power supply has five electrical terminals on the back and four on the front. Two coolant fluid lines are also located at the front (see Figure 3). Four pairs of *captive* fasteners hold the unit on the airframe. They are said to be captive because they remain attached to the unit after being unscrewed.

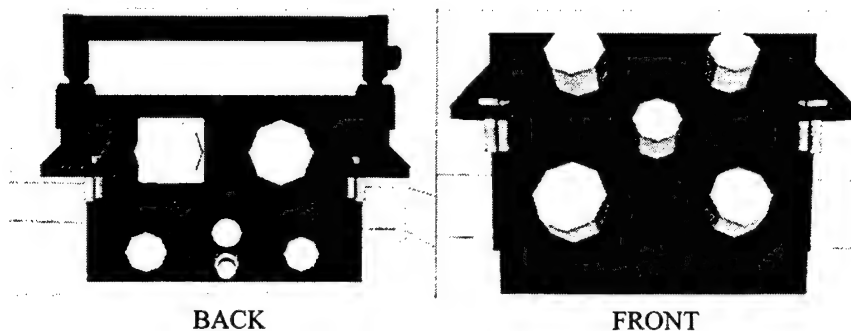


Figure 3. Front and back view of the E.W. power supply.

We chose to demonstrate only a subset of the tasks described in the L.S.A. document. More precisely, we modeled all the sub-steps of step 100 except steps 100d and 100f (installing protective devices) that are omitted for brevity.

The technician must disconnect the cables attached to the unit via their respective connectors, unbolt the unit from the airframe and extract it from the weapon bay. The corresponding technical orders do not specify a disconnection order among the groups of connectors located on the front and back of the unit (see 100b and 100c in 2.11). It will be up to the technician to choose the right disconnection order (see Figure 4).

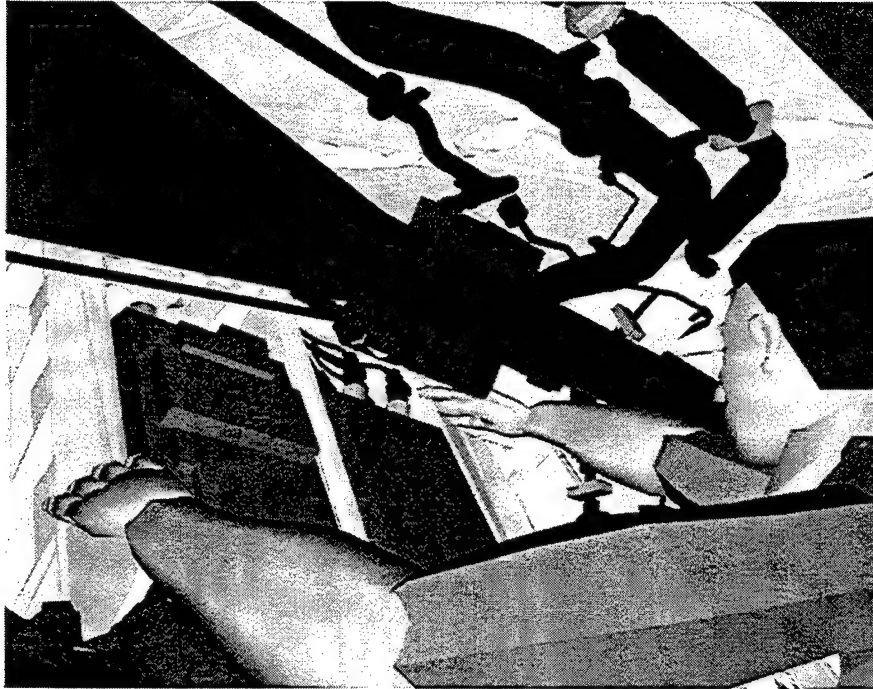


Figure 4. Jack reaching for the connectors on top of the power supply.

The only tool required is a wrench for the fasteners.

The LSA narration contains two warnings. The first one pertains to the danger of disconnecting powered electric cables. The second warns against the danger of prolonged exposure to the liquid coolant.

2.4 System Architecture

The ATOV demonstration environment is built on top of the EAI Jack software. It provides scene management, rendering, modeling, and graphical user interface (GUI) services. The PAR and ATOV specific components are plug-ins. Jack loads them automatically or upon user request. The ATOV plug-in deploys the user interface (UI) from which the demonstration scenarios are run.

The ATOV specific components are the agent controlling the virtual human, the motion generators for the primitive actions, the geometry observers, collision detection code, and the user interface. They are written in C++, except the UI, which is a Tcl-Tk script.

The user interface controls the agent and monitors its progress through a given task sequence.

The PAR modules are the PAR engine and the PAR simulator (simulation engine). The PAR engine executes the agent's high-level actions. The PAR simulator runs the environmental model consisting of physics-based and hazard models.

2.5 Environment Modeling

We used the CAD files provided by Lockheed-Martin to build the simulation environment. Because of the sheer amount of polygons contained in these files we were only able to retain a small subset of the geometry. This minimal environment contains the two bulkheads holding the power unit, a simplified model of the doors, the connectors and their respective cables. The technician is standing on an elevated work area in order to reach the top connectors at the back of the unit.

The maintainability study established that the missile launcher over-restricts the technician's access to the top of the power unit (see Figure 5). For the purposes of the study, we removed this component from our environment to improve overall accessibility and visibility. In a real world maintenance analysis, the impact of the missile launcher would have to be assessed for its impact on maintainer performance.

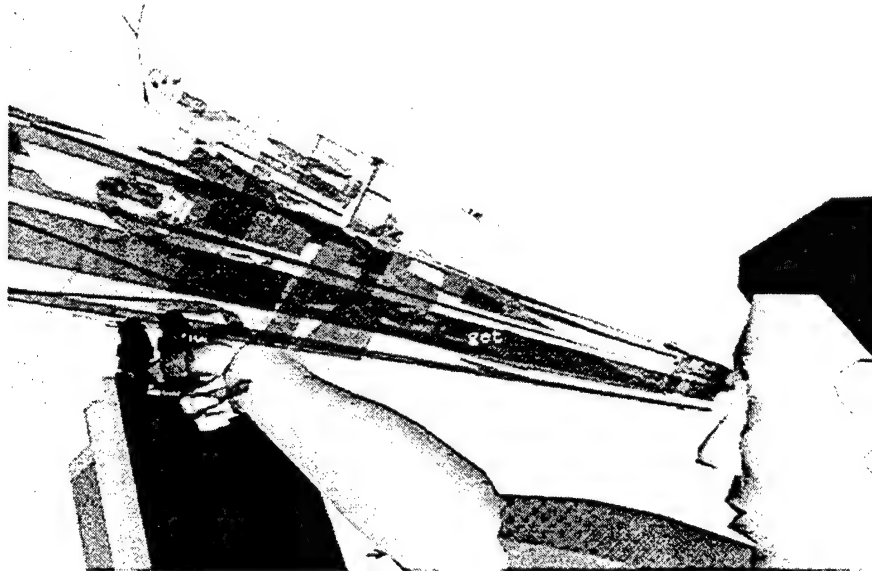


Figure 5. The missile launcher (in transparent brown) over-restricts the technician's access.

Each element of the scene has geometric and PAR models. Additional C++ and Tcl code was developed to provide low level functionalities and interfacing between the software components of the system. This includes collision detection routines. Collision detection is enabled between the virtual human's hand and the connectors.

2.6 Agents and Physical Objects in PAR

The virtual human is a PAR *agent*. Unlike a regular PAR object, an agent can perform PAR actions. For the purpose of this demonstration we created a PAR model for each physical object interacting with the virtual technician. The execution of the task affects the state of most objects. Each PAR object model has a **Status** field representing this state (

Table 5). Depending on its value the status of an object will trigger various sub-actions during the execution of a PAR action. Conversely, some PAR actions will update the status value as an effect of their execution.

<i>Name</i>	<i>Quantity</i>	<i>Status</i>
Power_Unit	1	InPlace / Supported / Extracted
Handle	1	Stowed / Transit / Extended
Connector	9	Connected / Disconnected
Fastener	8	Locked / Released

Table 5. Physical objects in PAR and associated status

2.7 Parametric Action Representation

We modeled the following steps of the LSA record:

- 100a Rotate the handle at the base of the unit.
- 100b Disconnect the five top electric connectors.
- 100c Disconnect the four bottom electric connectors.
- 100e Disconnect the two coolant lines.
- 100g Unbolt the height bolts retaining the power supply to the airframe and support it accordingly, and remove it.

As stated in 2.3, steps 100d and 100f are omitted.

The instructions warn the technician against the toxicity of the liquid coolant that might drain when the coolant lines are removed. OSHA regulations require that latex-like gloves and protective glasses must be worn at that time. We used the PAR simulator to model the contamination of the hydraulic connectors.

2.7.1 Connector Disconnection

There are three types of electric connectors. We will call them type I, type II, and type III. Five type I connectors are located on top of the unit. Two pairs of type II and III (one each) are located at the bottom of the unit. Two hydraulic connectors are also located on this side of the unit. The connectors have a cylindrical shape whose main axis corresponds to their mating direction (mating axis).

We abstracted the disconnection action as a single-handed reach followed by a grasp and a translation of the connector along its mating direction away from the unit. The actual disconnection would require the technician to rotate the sleeve of the connector to free it from the unit. Releasing the connector and withdrawing the hand complete the action.

The PAR disconnect action is of the form `Disconnect(agent, object, side)` where:

agent is the parameter for the agent performing the action,

object is the disconnected object,

side identifies which arm the agent should use for the action.

The action termination's condition occurs when the status of the object is set to **Disconnected**. If this condition holds true when the action is initiated, then it terminates immediately. Otherwise, it sets the status to **Disconnected** after executing the following sub-action sequence:

```
Approach(agent, object, side),
ShapeHand(agent, side, t_1_grasp),
Hold(agent, object, side, Hold),
Reach(agent, object, Park, side),
Hold(agent, object, side, Release),
ShapeHand(agent, side, t_1_shape), and
Reach(agent, object, Clear, side).
```

The Approach action is meant to prepare the *agent's* hand to grasp the *object*. It is a combination of a `Reach(agent, object, Clear, side)` and a `ShapeHand(agent, side, t_1_shape)`.

When Disconnect is executed the virtual technician reaches for the connector, grasps it, moves it away from the power unit, releases it and withdraws its hand. Since we do not model rigid body dynamics, the connector, released in free space, remains immobile. We found this solution an acceptable abstraction for the task for putting away the connectors and their cables at a location where they will not fall nor interfere with the technician's work.

ShapeHand, Hold, and Reach are primitive actions. The **Hold** action attaches and detaches an *object* to the hand of an *agent*. **ShapeHand** shapes the hand of the *agent*. We used two predefined hand shapes *t_1_shape*, and *t_1_grasp* (see Figure 6). The first one pre-shapes the hand to grasp a connector. The second one corresponds to the grasped position itself (see below).

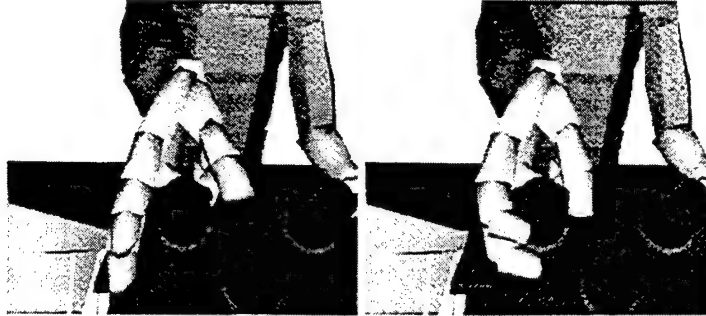


Figure 6. *t_1_shape* and *t_1_grasp* hand shapes

It is important to note that Jack can automatically grasp an object by closing its hand on it. However, at the time of this work, this feature was not available through the Jack Toolkit API.

The three primitive actions, **ShapeHand**, **Hold**, and **Reach** are, the basis of the PAR vocabulary for encoding actions involving arm motions.

2.7.2 Point-to-Point Reach

The virtual human can reach for a given point in space with the *Reach(agent, object, purpose, side)* action. Its parameters are as follows:

- agent* is the parameter for the reaching agent,
- object* is the object to be reached,
- purpose* indicates the intent of the agent,
- side* identifies which arm the agent should use to reach.

Reach uses a recently developed fast inverse kinematics routine, **IKAN** (Tolani,99), to generate the appropriate arm motion between the arm's current position and the desired one. This system works by designating an end-effector site (location) on the arm and a target site on the object to be reached. **IKAN** computes set of arm joint angles that places the end-effector site on the target site. Then the **Reach** generates the motion by interpolating the arm's joint angles between its initial position and **IKAN**'s solution. As the motion is rendered, the end-effector site rendezvous with the target site.

The actual target site depends on the nature of *agent*, *object*, and *purpose*. For the virtual human and a given connector we defined four purposes: **Approach**, **Disconnect**, **Park**, and **Clear** (see Figure 7). They are respectively used to bring the technician's hand near the connector, translate the hand to its grasping location, move the connector away from the power unit along its mating axis, and withdraw the hand from the connector.

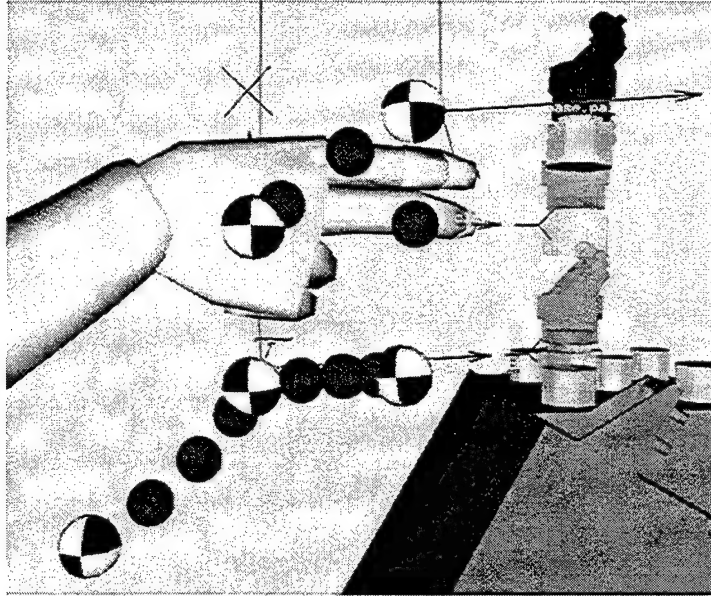


Figure 7. The four steps of a Disconnect action

For each purpose we defined a pair of end-effector and target sites (see Figure 8). The target sites are defined relative to the *object*.

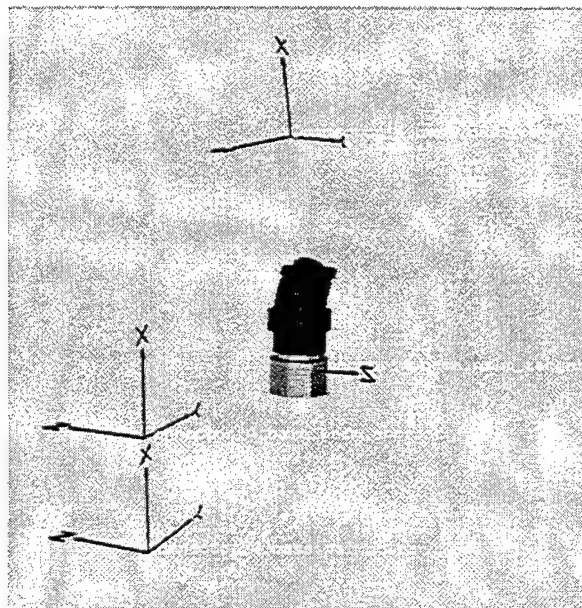


Figure 8. Connector annotated with target sites

In the F-22 environment, the technician must bend his spine to reach over the power unit. Since IKAN deals only with arms we added a separate controller to bend Jack's spine and increase its reach volume. Each Reach provides a point-to-point control of the virtual human's arm. When the agent executes multiple reaches in sequence (as in Disconnect) each reach is executed independently from the previous and following one. The arm stops at the target site and leaves right away. Although it is anatomically correct, the overall arm motion is piecewise linear and looks robotic. A motion-planning module would blend consecutive point-to-point reaches into more natural continuous reach. More importantly, a motion planner would dynamically select intermediate waypoints to reach for an object, and grasp it at the right location.

2.7.3 Power Supply Extraction

Although extracting the power unit is concisely described by "...and remove the power supply." in task 100g, it is very complex and requires a lot of common sense reasoning from the technician. Indeed he must first lift the front part of the unit by the handle to make its bottom face accessible (while the top of the unit remains on the airframe). Then he applies his second hand on the bottom to support the whole weight of the unit as he lifts its top part away from the airframe (Figure 9).



Figure 9. Power supply extraction

We encoded task 100g with the action *Extract(agent, power_supply)*. Its preparatory specifications require that the eight fasteners must be released before the extraction must take place. This means that if some of the fasteners are locked when *Extract* starts, the virtual technician will "sub-goal" into releasing them before proceeding. This covers the beginning of task 100g instructing the technician to release the fasteners. This "sub-goaling" is also a way of encoding the technician's common sense since he knows that the fasteners of a component must be released prior to removing the component itself. This is also a good example demonstrating the usefulness of a disassembly planner.

We also treated supporting the power unit as a preparatory specification. Since these specifications are checked and satisfied in sequence, we inserted a requirement for supporting the unit before the two last fasteners are released (see Table 6).

Condition	Action
<i>power_supply.Fastener_1.Status = Released</i>	<i>Release(agent, power_supply.Fastener_1,Right)</i>
<i>power_supply.Fastener_2.Status = Released</i>	<i>Release(agent, power_supply.Fastener_2,Right)</i>
<i>power_supply.Fastener_3.Status = Released</i>	<i>Release(agent, power_supply.Fastener_3,Right)</i>
<i>power_supply.Fastener_4.Status = Released</i>	<i>Release(agent, power_supply.Fastener_4,Right)</i>
<i>power_supply.Fastener_5.Status = Released</i>	<i>Release(agent, power_supply.Fastener_5,Right)</i>
<i>power_supply.Fastener_6.Status = Released</i>	<i>Release(agent, power_supply.Fastener_6,Right)</i>
<i>power_supply.Status = Supported</i>	<i>Support(agent, power_supply,Left)</i>
<i>power_supply.Fastener_7.Status = Released</i>	<i>Release(agent, power_supply.Fastener_7,Right)</i>

<i>Condition</i>	<i>Action</i>
<i>power_supply.Fastener_8.Status = Released</i>	<i>Release(agent, power_supply.Fastener_8,Right)</i>

Table 6. Preparatory specifications for the Extract action.

2.7.4 Other Complex Actions

2.7.4.1 Release Fastener

The virtual technician releases a fastener with *Release(agent, fastener, side)*. Although this action is similar to *Reach* it requires the use of a wrench to actually release the fasteners. We did not model the act of fetching the wrench. A preparatory specification puts a wrench in the agent's hand if necessary. The primitive action associated with the wrench action is *Use(agent, wrench, fastener, purpose)*, where purpose is either **Lock** or **Release**.

2.8 Failures

We modeled three types of failures to include:

- **Out-of-Reach:** the virtual human cannot reach.
- **Collision:** the hand of the agent or whatever solid it is holding collides with the environment while moving¹.
- **Fastened-Unit:** the technician attempts to extract the power supply while it is still fastened to the airframe.

The out-of-reach and the fastened-unit failures cause all agent activities to halt. In Figure 10, the front connector was successfully removed while the back connector is out of reach. On the other hand, the collision is handled differently depending on the purpose of the reach. If the collision occurs while the



Figure 10. Out of Reach Connector

¹ As mentioned in 2.5 we only handled collisions between the virtual human and the connectors.

agent reaches for a connector then the Reach action fails along with the chain of super-actions that triggered it. For example, if the agent collides with a connector while reaching for another one during a Disconnect, both Disconnect and Reach abort. On the other hand, if the collision occurs while the hand is moving away from an object or is moving a connector the Reach terminates successfully. Figure 11 shows how collision detection while they are moved away from the power unit staggers them in space.

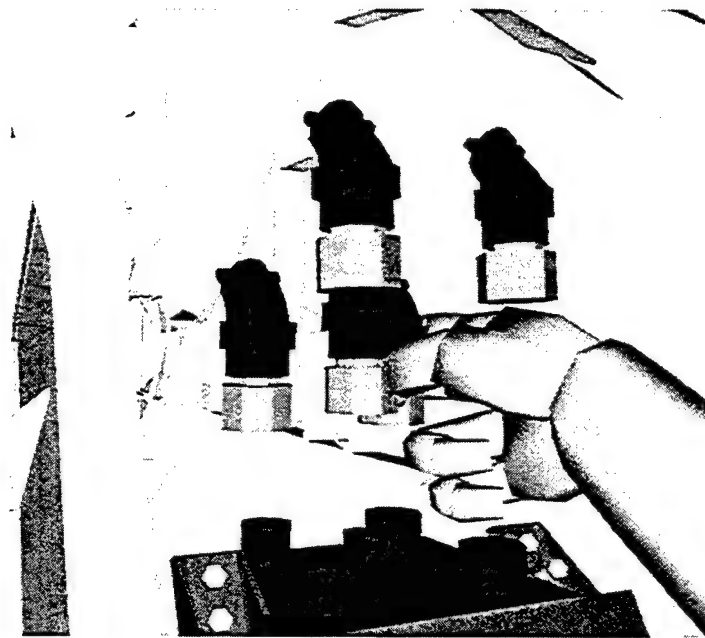


Figure 11. Staggered Connectors

The inverse kinematics solver of Reach generates the out-of-reach failure when its solution comes short of reaching for the target site. The fast collision detection algorithm, V-Collide (Hudson,97), reports interpenetrating objects to the active Reach actions. Depending on their purpose each action decides to terminate, or to report collision failure and abort. Finally, the PAR simulator maintains an abstract assembly graph of the power supply attached to the airframe (see section 2.9.2). It issues a fastened-unit failure when the agent attempts to extract the power supply when one of its fasteners is still locked.

2.9 Multi-Modeling

The power supply, the connectors, the fasteners, the handle, and the airframe are modeled in multiple domains. They all have geometric and PAR models. The fasteners, the hydraulic connectors, the power unit, and the airframe are modeled in the PAR simulator. It models the assembly graph relating the fasteners, the power supply, and the airframe. It also models the contaminating effect of the cooling fluid on the hydraulic connectors (environmental model).

Validating a maintenance task does not call for direct user manipulation since a virtual human does the work. However, setting up or adjusting the input conditions of the task requires user geometry manipulations. The user might also want to perform a few maintenance tasks manually. Therefore, it is important to maintain a uniform interface for both user and virtual technician.

We applied our uniform interaction interface approach (see Section 1.2.1) to these multi-models.

2.9.1 Geometry Observers

The status of most objects is geometrically driven. For example, the handle is **Stowed** or **Extended** when the angle of its joint is respectively 0° and 180° . Within Jack, the joint can be directly manipulated by the

user or procedurally via the motion generator of the MoveHandle primitive action. However, the effect of these manipulations (status update) in PAR should be the same.

We solved this geometry/PAR synchronization problem with *observers*. These objects are activated each time the state of their Subject changes. In this case the subject is the handle's joint. The observer updates the handle's PAR status. We used the same solution for the fasteners.

Observers have one extra benefit. They are updated the first time the environment is displayed. This means that the PAR attributes of their subjects are automatically initialized to their proper value.

2.9.2 Assembly Graph

The maintenance procedure calls for the removal of a power unit from a F-22 airframe. This component is physically attached by fasteners (bolts) to the aircraft. We chose to simulate their function by reporting an error if the virtual technician tries to move the unit while one of the fasteners is still Locked.

The fastening effect of bolts comes from their shape. Thanks to their helicoidal geometry they can screw into a tapped bore and hold down a part if torqued properly. This geometrical property is not practical to simulate in a virtual environment. Instead, we use an assembly graph (see Figure 12) to model the power supply assembly.

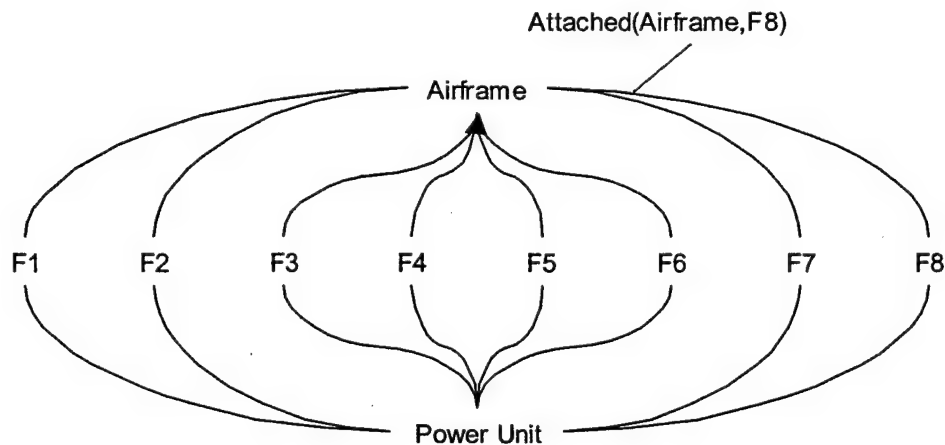


Figure 12. Assembly Graph

The vertices of the graph are the parts. The edges represent an assembly relationship between two parts. For this demonstration we only use one relation, $\text{Attached}(p1, p2)$, to denote that two parts $p1$ and $p2$ are attached. A second unary predicate, $\text{Grounded}(p)$, indicates that a part p is linked to the ground, possibly by a chain of attached parts. A sub-assembly cannot be removed when one of its parts is grounded.

The Extract action checks whether the power supply is grounded before moving it. If it is the case it will fail. Similarly, locking or releasing the fasteners asserts or retracts their Attached relationship with the airframe. The fasteners' observers directly perform this update.

The assembly graph is maintained within the PAR simulator. We use the logic programming features of the simulator's modeling language to directly encode the Attached and Grounded relations (the models are in 0).

The parts of the assembly are the airframe, the power supply, and its eight fasteners. The airframe is permanently grounded. Since the fasteners are captive they are permanently attached to the power supply.

2.9.3 Environmental Model

The notices of the LSA record warn the technician to use adequate protection against the toxic cooling fluid and to wipe off any spills. We created a simple environmental model to represent the areas contaminated by

the fluid. These areas are the connectors of the hydraulic lines. Green transparent spheres indicate contamination.

Hydraulic connectors and the fluid identity are modeled in the PAR simulator. The fluid has a toxic attribute to indicate whether it is dangerous; in this case it is. The `Connected(c1,c2)` indicates that two connectors are connected. A `Carry(c,s)` relation denotes that the connector `c` carries the fluid `s`.

The contamination object, represented by a green transparent sphere, uses an instantiation rule to detect connectors exposed to the environment (i.e. disconnected) carrying a toxic fluid (see Figure 13). The sphere remains attached to the contaminated object.

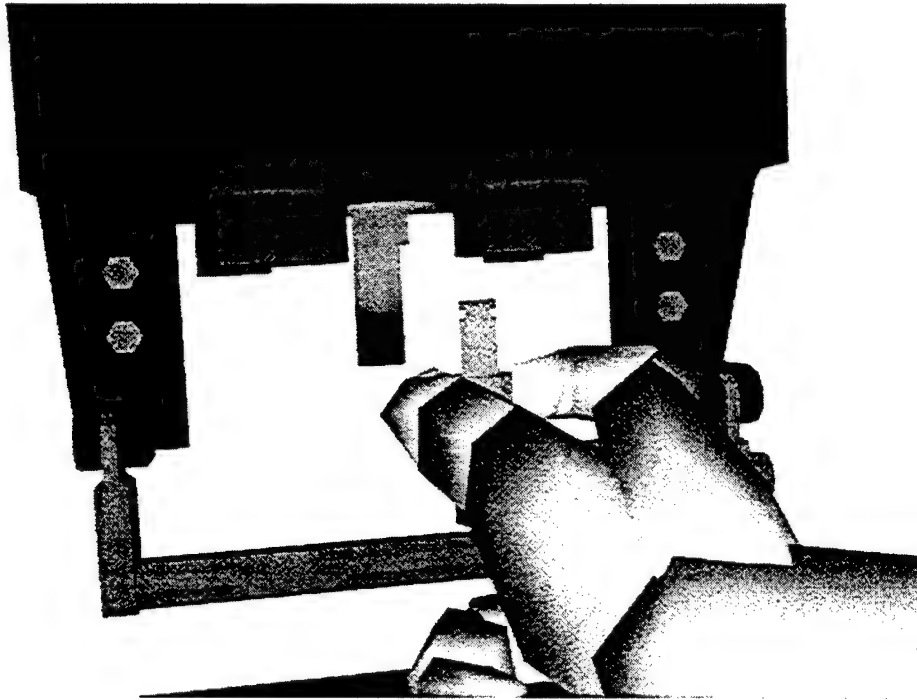


Figure 13. Warning spheres indicating contaminated hydraulic disconnects

The environment model (see 0 for models) is fairly general since it is able to flag any contaminated connector (see Figure 13). Although it is not featured in this demonstration, a standing order to wipe off contaminated areas could be given to the virtual technician.

2.10 Scenarios

We designed four scenarios to demonstrate the capacities of the ATOV framework. The first three showcase a specific type of failure as described in section 2.8. These scenarios represent the successive attempts of a TO author simulating and correcting a maintenance procedure until it completes successfully.

The first scenario shows an `OutOfReach` failure. The agent is able to start the procedure (i.e. extend the handle) but since he is standing on the ground he fails to reach the first connector on top of the power supply. The scenario ends with an error message being printed on the display.

In the second scenario the technician is standing on an elevated platform. However his first reach fails again. This time the PAR instruction required it to reach for a connector at the back of the top of the unit. The failure occurs when its hand collides with the connector in front of the one it was reaching for. The collision error is reported by the user interface.

The third scenario starts after the order of the connector's disconnection has been corrected to reflect accessibility. The procedure runs until the technician tries to extract the power supply. The `Extract` action it is using is deprived of the preparatory specifications requiring it to release the fasteners. The assembly

graph model triggers a failure when an attempt to move the unit is made. The user interface reports the error.

The Extract action is corrected for the fourth attempt. Indeed the fourth scenario completes successfully and the user interface reports success.

During the third and fourth attempt the environment model flags the hydraulic terminals with contamination markers as described in Section 2.9.3.

2.11 Conclusions and Recommendations

Automated TO validation is both a useful concept and a challenging research area. To fully realize the potential of ATOV, however, a number of software components must be developed and integrated into the TO authoring and validation process. We have previously noted that advances in Product Data Models (PDMs) will greatly aide the integration of annotated CAD models within the TO authoring environment. Additional human form modeling software features such as confined reach space planning will be essential to more automated task analyses. Continued progress in moving high performance graphics onto the TO author's desktop will also be advantageous. Our goal must not be to redefine the role of the TO author as an animator, but rather to provide the author with better tools to interpret, simulate, and validate TOs as written.

Whether integrated into an IETM authoring environment or available as plug-ins or additional modules for design evaluation systems, automated TO validation still requires fundamental research work. Some of the research needs include:

- Translators that can automatically extract the important actions and PAR characteristics from TO text.
- Development of general definitions of PARs for maintenance tasks. These PARs must reference assemblies and parts in a fashion consistent with part nomenclature and PDM structures.
- Execution of motion generators for complex tasks such as confined reach space planning.
- Disassembly planners to insure geometric and spatial access can be used to help sequence maintainer actions; but such planners must be extended to use human reach planning and maintenance access solids as well.
- Exhaustive analyses rather than just interactive testing to insure that maintenance tasks can be done by a suitable maintainer population with and without special clothing or protective gear.
- Failure reports with sufficient detail to allow visualization of problem areas when exhaustive searches fail. Our demonstration shows that task failures can be detected by monitoring PAR execution and can be illustrated in 3D graphics suitable for task exposition or training.
- Interactive component connection and function models to properly mitigate hazards and expose cautions arising from incorrect disassembly procedures or non-geometric operational hazards such as heat, pressure, or radiation.
- User interfaces that integrate PDM databases and IETM or similar TO authoring systems with PAR simulation and visualizations.
- Task analysis results that can be saved and replayed for different sized maintainers for training purposes.

In our opinion, the most critical research need is constrained reach space planning. A general, effective, and efficient constrained reach planner may be difficult to develop. While background work in robotics is relevant, human bodies present many more degrees of freedom, vary in their available strength, can brace against other stable structures, and have a mobile base of support. Maintainer population anthropometric variability and soft constraints presented by the human body's physiology are additional challenges for a reach planner.

While this study has not been optimistic about the role and efficacy of Virtual Reality in ATOV, it is possible that whole body haptic sensing could compete in both speed and versatility with a non-interactive reach space planner. By using a whole body suit with haptic pressure actuators, one may be able to use the subject's skill and experience in maintenance situations to establish reasonable reach postures and approaches. The subject would be informed via the haptic actuators of unacceptable penetrations with the CAD modeled environment. While one would still miss the supports and physical surfaces of the real environment, it might be the only way to approximately assess effective strength in an awkward pose. Ultimately, however, we feel that more automated task analysis components are the only sure path toward substituting virtual models for physical systems.

Appendices

A1 LSA Task Narrative (Sample)

File: In-Work End Item: F22A LCN: A991215PS1 Task Code: WGOFEAA
(U) R/I L FWD PWR SPLY (FOM)

Total Elapsed Time = 17.4 min. Total MM = 17.4 min. Mean Crew Size = 1.00

TASK NARRATIVE	PID	SSC	MM	ET	AREA	5	10	15	20
10. VERIFY ACFT SAFE FOR MAINTENANCE				0.00					
<div>COMMENT</div> <div>A002000/00/AGOFCAA</div>									
(REF) 20. (U) CONNECT PMA (L MAIN W/W)	A	2A353N	0.50	0.50		0.5			
(REF) 25. OPEN LEFT SIDE WEAPONS BAY DOORS	B	2A3Z3N	2.00	2.00	0.5	2.5			
30. (U) PERSONNEL RECOMMENDED: ONE				0.00					
40. (U) SAFETY CONDITIONS: NONE				0.00					
100. (U) REMOVE LEFT FWD PWR SPLY	C	2A250	3.20	3.20	2.5	5.7			
100a. (U) Press release button on handle and rotate to extended position.									
<div>WARNING</div> <div>(U) Electrical power shall be off before connecting or disconnecting electrical connectors. Failure to comply may result in injury to personnel and/or damage to aircraft or equipment.</div>									
100b. (U) Disconnect 991215P2, 991215P3, and 991215P8.									
100c. (U) Disconnect 991215P1, 991215P4, 991215P5, 991215P6 and 991215P7.									
100d. (U) Install protective devices on electrical disconnects.									
<div>WARNING</div> <div>(U) Polyalphaolefin heat transfer fluid is toxic. OSHA regulations require the use of appropriate Personnel Protective Equipment (PPE) suited to application methods. Check input conditions for local Bioenvironmental Engineering PPE recommendations. Avoid repeated or prolonged contact. Failure to comply may result in injury and/or illness to personnel.</div>									
100e. (U) Disconnect two coolant lines and absorb any excess polyalphaolefin fluid with a lint free cloth.									
100f. (U) Install protective devices on coolant disconnects.									
100g. (U) Support power supply, release eight captive fastener, and remove power supply.									
(REF) 101. (U) INSP L FWD PWR SUPPLY RAIL	C	2A250	1.50	1.50		5.7	7.2		

A2 Assembly Graph Models

We build assembly graphs with two layers of models. The first one is general. It captures the basic concepts underlying a graph. The second layer is task specific. It consists of specialized models extending the general models to apply the assembly graph concepts to the parts the virtual human actually manipulates.

A2.1 General Models

We build assembly graphs with the three following models. The `AF_Part` and `AF_Attached` respectively correspond to the nodes and edges of a graph. `AF_Grounded` is a predicate to identify the grounded state of a part. This predicate is transitive. If a part is grounded then all the parts attached to it, through an `AF_Attached` instance, are also grounded. This transitivity property is modeled by the rule in `AF_Part`.

```
// Relation indicating that a part cannot be removed from the main
// assembly
// AF_Grounded.frag
```

```
fragment AF_Grounded(AF_Part p)
{
    enter{ p.grounded := true;};
    exit{ p.grounded := false;};
};
```

```
// Assembly part fragment
// AF_Part.frag
```

```
fragment AF_Part
    extern "AF_Shell.so"
{
    bool grounded; // Flag indicating whether the AF_Part is grounded
    String name; // Name of the AF_Part

    rule AF_Part other, AF_Attached(other,self) a :
        => clause AF_Attached(other,self) and
            AF_Grounded(other)=>AF_Grounded(self);

    create(String s1)
    {
        init(s1);
    };

    void init(String s1)
    {
        name := s1;
    };
};
```

```
// Relation indicating that two assembly parts are attached
// AF_Attached.frag
```

```
fragment AF_Attached(AF_Part p1, AF_Part p2)
{
    // Reflexivity of the attached relation
    clause AF_Attached(p1,p2)=>AF_Attached(p2,p1);
};
```

A2.2 Maintenance Task Specific Models

For the purpose of the power unit removal, we specialized the AF_Part model into models capturing the specificities of the F22 airframe, the fasteners and the power unit itself. In particular, the AF_F22 model is grounded by default. Furthermore, since a fastener is a captive part of the power unit, the AF_Fastener model is initialized with an AF_PowerUnit instance. Finally, the clause in AF_Fastener permanently attaches a fastener to its power unit.

```
// Power supply object
// AF_F22.frag
```

```
fragment AF_F22
  inherit AF_Part
{
  create(String s1)
  {
    init(s1);
    assert AF_Grounded(self);
  };
};
```

```
// Power supply object
// AF_PowerSupply.frag
```

```
fragment AF_PowerSupply
  inherit AF_Part
{
  create(String s1)
  {
    init(s1);
  };
};
```

```
// Fastener object
// AF_Fastener.frag
```

```
fragment AF_Fastener
  inherit AF_Part
{
  AF_PowerSupply ps; // Power supply to which the AF_Fastener belongs
  AF_Part part; // AF_Part to which the AF_Fastener is fastened

  create(String s1, AF_PowerSupply o, AF_Part p)
  {
    init(s1);
    ps := o;
    part := p;
  };

  clause AF_Attached(self, ps);
  // Attaches the AF_Fastener to the AF_PowerSupply
};
```

A3 Environmental Model

An *environmental model* is a collection of models whose overall function is to describe how a device assembly affects its surrounding environment. For example, the environmental model can have a model for leaks if the device is liable to release liquids in the environment. We created an environmental model to indicate terminals contaminated by liquid coolant (see 2.9.3).

A3.1 General Models

The general models below represent the notion of a liquid being carried by a hydraulic terminal. If the liquid is toxic then a terminal carrying it and exposed to the environment (i.e., it is disconnected) will be flagged as contaminated.

```
// Models some of the properties of liquid materials
// AF_Liquid.frag

fragment AF_Liquid
{
    bool toxic; // indicates that this liquid is toxic
};

// Fragment to model hydraulic terminals
// AF_Terminal.frag

fragment AF_Terminal
{
    bool connected; // Flag indicating that a terminal is connected
};

// Models the fact a that a terminal carries a given liquid
// AF_Carries.frag

fragment AF_Carries(AF_Terminal term, AF_Liquid liq)
{
    rule : liq.toxic and not term.connected
        => clause AF_Contaminated(term,liq);
    // The rule creates an AF_Contaminated for each non
    // connected AF_Terminal carrying a toxic liquid
};

// Contamination predicate associates a terminal and a toxic liquid
// AF_Contaminated.frag

fragment AF_Contaminated(AF_Terminal term, AF_Liquid liq);
```

A3.2 Maintenance Task Specific Models

The following task specific models extend the general models above to introduce the notion of contamination in our scenario (see 2.1). In particular, we define a model for the liquid coolant. We model the connector by extending the AF_Part (defined in 0) and the AF_Terminal models. The AF_Warning model creates a transparent sphere marker in the scene for each AF_Contaminated instance (see Figure 13).

AF_Terminal and AF_Warning use external C++ code to interface directly with geometric figures representing them in Jack.

```
// Models the coolant liquid
// AF_Coolant.frag

fragment AF_Coolant
  inherit AF_Liquid
{
  create()
  {
    toxic := true;
  };
};

// Connector object
// AF_Connector.frag

fragment AF_Connector
  inherit AF_Part, AF_Terminal
  extern "AF_Shell.so"
{
  create(String s1)
  {
    init(s1);
  };
};

// Contamination visual warning
// AF_Warning.frag

fragment AF_Warning(AF_Contaminated c)
  instantiate true
  extern "AF_Shell.so"
{
  AF_Terminal term := c.term;
};
```

References

- Badler,00:** Badler, N., Bindiganavale, R., Allbeck, J., Schuler, W., Zhao, L. and Palmer, M. Parameterized Action Representation. *Embodied Conversational Agents*, The MIT Press, Cambridge MA, 2000.
- Erignac,01:** Erignac, C. *Interactive Semi-Qualitative Simulation*, Ph.D. Thesis, University of Pennsylvania, 2001.
- Harel,87:** Harel, D. Statecharts: A visual formulation for complex systems. *Science of Computer Programming* 8(3), pp. 231-274, June 1987.
- Bindi.,00** Bindiganavale, R. *Building Parameterized Actions from Observations*, Technical Report, MS-CIS-00-17, Ph.D. Thesis, University of Pennsylvania, 2000
- Barron,97:** Barron, M. and Abshire, K., Design maintainability demonstration for F-22 avionics. Lockheed-Martin Aeronautical Systems Company, United States Air Force, 1997.
- Tolani,99:** Tolani, D., Goswami, A. and Badler, N. Real-time inverse kinematics techniques for anthropomorphic limbs, *Graphical Models* 62, pp. 353-388, 2000.
- Hudson,97:** Hudson, T., Lin, M., Cohen, J., Gottschalk, S. and Manocha, D. V-COLLIDE: Accelerated collision detection for VRML. In *Proceedings of VRML '97*, 1997.